

Overview

With the introduction of [Azure DevOps](#), Microsoft is offering developers a new continuous integration/continuous delivery (CI/CD) service called [Azure Pipelines](#) that enables you to continuously build, test, and deploy to any platform or cloud. It has cloud-hosted agents for Linux, macOS, and Windows; powerful workflows with native container support; and flexible deployments to Kubernetes, VMs, and serverless environments.

Azure Pipelines provides unlimited CI/CD minutes and 10 parallel jobs to every GitHub open source project for free. All open source projects run on the same infrastructure that our paying customers use. That means you'll have the same fast performance and high quality of service. Many of the top open source projects are already using Azure Pipelines for CI/CD, such as Atom, CPython, Pipenv, Tox, Visual Studio Code, and TypeScript-and the list is growing every day.

In this lab, you'll see how easy it is to set up Azure Pipelines with your GitHub projects and how you can start seeing benefits immediately.

Objectives

- Install Azure Pipelines from the GitHub Marketplace.
- Integrate a GitHub project with an Azure DevOps pipeline.
- Track pull requests through the pipeline.

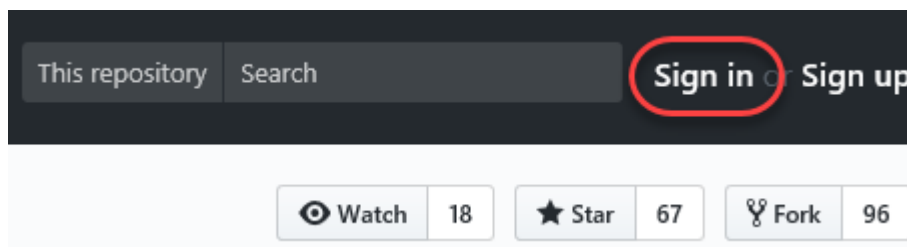
Prerequisites

- An Azure DevOps account from <https://dev.azure.com>.
- A GitHub account from <https://github.com>.

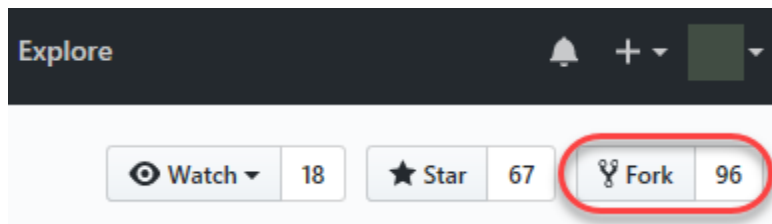
Exercise 1: Getting started with Azure Pipelines

Task 1: Forking a GitHub repo and installing Azure Pipelines

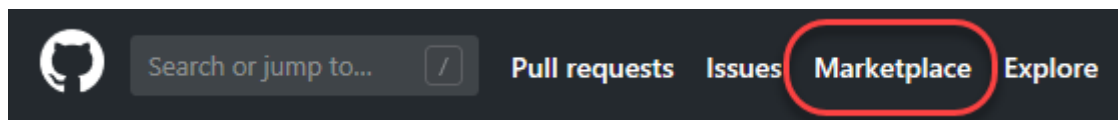
1. Navigate to <https://github.com/calculator-demo/calculator>. This is the baseline project we will fork and use for this lab.
2. If you're not already signed in to GitHub, sign in now.



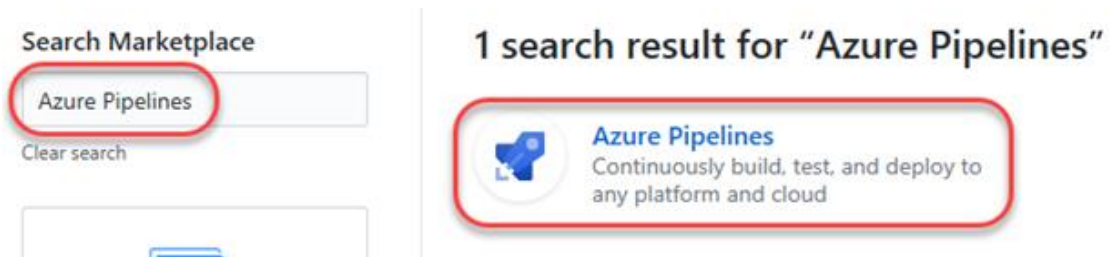
3. Click **Fork** to fork the repository to your own account.



4. If prompted, select an account to fork the repository into.
5. The **GitHub Marketplace** provides a variety of tools from Microsoft and 3rd parties that help you extend your project workflows. Click **Marketplace** from the top navigation to visit it.



6. Search for “**Azure Pipelines**” and select the result.



7. Click **Read more**.

Azure Pipelines

Set up a plan

Continuously build, test, and deploy to any platform and cloud

Azure Pipelines offers cloud-hosted pipelines for Linux, macOS, and Windows with 10 free parallel jobs and unlimited minutes for open source projects.

[Read more...](#)

8. Take a moment to read through the benefits of Azure Pipelines.

🔧 Any language, platform, and cloud

Build, test, and deploy Node.js, Python, Java, PHP, Ruby, Go, C/C++, C#, Android, and iOS apps. Run in parallel on Linux, macOS, and Windows. Deploy to cloud providers like Azure, AWS, and GCP. Distribute mobile apps through beta channels and app stores.

🐳 Native container support

Create new containers with ease and push them to any registry. Deploy containers to independent hosts or Kubernetes.

🔗 Advanced workflows and features

Easy build chaining and multi-phased builds. Support for YAML, test integration, release gates, reporting, and more.

🔑 Extensible

Use a range of build, test, and deployment tasks built by the community – hundreds of extensions from Slack to SonarCloud. You can even deploy from other CI systems, like Jenkins. Webhooks and REST APIs help you integrate.

💙 Free, to you from Azure Pipelines

Free cloud-hosted builds for public and private repositories.

9. The Azure Pipelines offering is free for anyone to use for public repositories, and free for a single build queue if you're using a private repository. Click **Install it for free**.

Pricing and setup

Free
Free for public and private repositories

\$0

Add parallel jobs
Add parallel jobs for private repositories

\$40
per parallel job / month

Azure Pipelines

Free

Free for public and private repositories

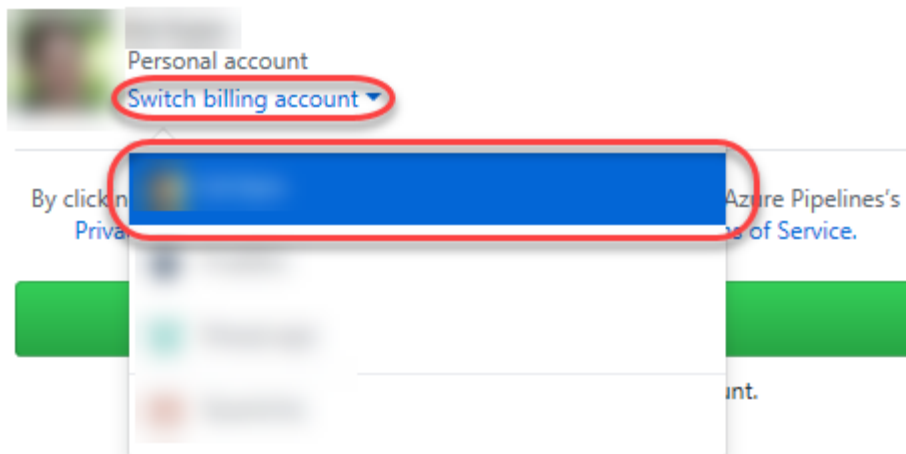
- ✓ Linux, macOS, and Windows
- ✓ 10 free parallel jobs for public repositories
- ✓ Unlimited minutes for public repositories
- ✓ 1 free parallel job for private repositories (1,800 minutes per month)

Install it for free

Next: Confirm your installation location.

10. If you have multiple **GitHub** accounts, select the one you forked the calculator to from the **Switch billing account** dropdown.

Billing information



11. Click **Complete order and begin installation**.

By clicking "Complete order and begin installation", you agree to Azure Pipelines's [Privacy Policy](#). You previously agreed to the [Marketplace Terms of Service](#).

Complete order and begin installation

Next: Authorize Azure Pipelines to access your account.

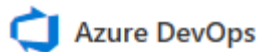
12. You have the option to specify repositories to include, but for the purposes of this lab, just include all of them. Note that Azure DevOps requires the listed set of permissions to fulfill its services. Click **Install**.

The screenshot shows the GitHub App permissions configuration interface. At the top, there are two radio button options: 'All repositories' (selected) and 'Only select repositories'. Below the 'Only select repositories' option is a button labeled 'Select repositories' with a dropdown arrow. A horizontal line separates the repository selection from the permissions section, which is headed '...with these permissions:'. Under this heading, there are three checked items: 'Write access to code', 'Read access to metadata', and 'Read and write access to checks, commit statuses, deployments, issues, and pull requests'. At the bottom, there are two buttons: a green 'Install' button (highlighted with a red rounded rectangle) and a blue 'Cancel' button. Below the buttons, a text line reads: 'Next: you'll be directed to the GitHub App's site to complete setup.'

13. You may be prompted to confirm your GitHub password to continue.
14. You may be prompted to log in to your Microsoft account. Make sure you're logged into the one associated with your Azure DevOps account.

Task 2: Configuring your Azure Pipelines project

1. You are now on the Azure DevOps site and need to set up your Azure Pipelines project. Select (or create) the **Azure DevOps organization** you would like to perform these builds under, as well as the Azure DevOps **project** from that organization you would like to use. Click **Continue**.



Setup your Azure Pipelines project

Select your Azure DevOps organization *

Create a new organization

Select a project *

Create a new project

Choosing **Continue** means that you agree to our [Terms of Service](#), [Privacy Statement](#), and [Code of Conduct](#).

Continue

2. Select the **calculator** project from GitHub to build as part of the pipeline.

Select a repository

Filter by keywords

Mine


/calculator fork

15 minutes ago

① Showing the most recently used repositories where you are a collaborator.
If you can't find a repository, make sure you [provide access](#).

3. Azure Pipelines will analyze your project in an attempt to determine if any existing templates would be a good fit. In this case, the recommended template is for **Node.js**, which is perfect for our needs. Some alternative templates are also suggested, although the recommended one is the best for this lab. Select it to continue.

Choose a template



Node.js

Build a general Node.js application with npm.

Recommended

- The build pipeline is defined as **YAML**, a markup syntax well-suited to defining processes like this because it allows you to manage the configuration of the pipeline like any other file in the repo. It's a pretty simple template that identifies the pool to pull a VM from for building, the process to install Node.js for building, and the actual build itself. Click **Save and run** to save the pipeline and queue a new build.

azure-pipelines.yml

Save and run

```
1  # Node.js
2  # Build a general Node.js application with npm.
3  # Add steps that analyze code, save build artifacts, deploy, and more:
4  # https://docs.microsoft.com/vsts/pipelines/languages/javascript
5
6  pool:
7    vmImage: 'Ubuntu 16.04'
8
9  steps:
10   - task: NodeTool@0
11     inputs:
12       versionSpec: '8.x'
13     displayName: 'Install Node.js'
14
15   - script: |
16     npm install
17     npm run build
18     displayName: 'npm install and build'
```

- For the purposes of this lab, you can commit this new file directly to the master branch. Click **Save and run**.

Save and run



Saving will commit `./azure-pipelines.yml` to the repository.

Set up CI with Azure Pipelines

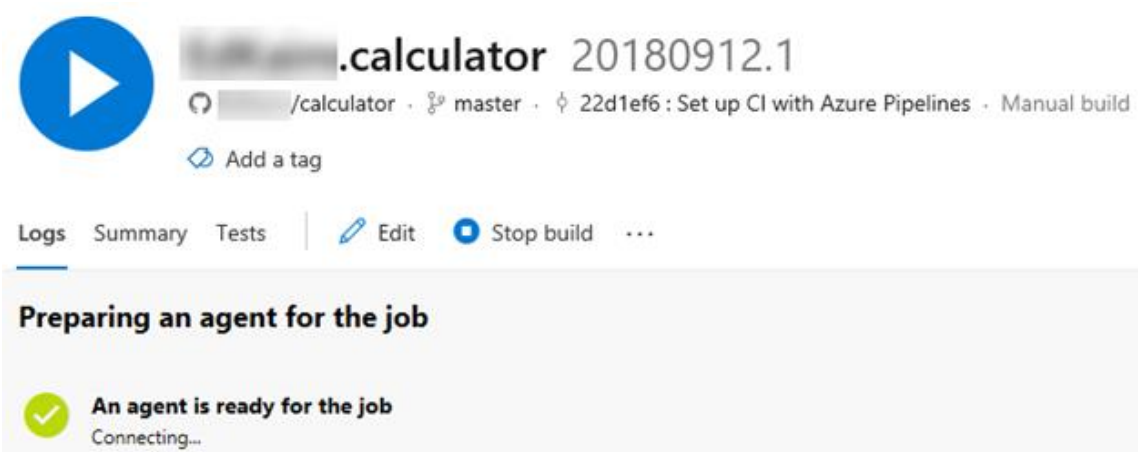
Add an optional description...

- ☒ Commit directly to the master branch.
- ☐ Create a new branch for this commit and start a pull request.

Cancel

Save and run

6. It will take a moment for the pipeline to complete. During this time it will configure the build agent, pull in the source from GitHub, and build it according to the pipeline definition.




.calculator 20180912.1

/calculator · master · 22d1ef6 : Set up CI with Azure Pipelines · Manual build

Add a tag

Logs Summary Tests | Edit Stop build ...

Preparing an agent for the job

 **An agent is ready for the job**
Connecting...

7. The build should complete successfully.



.calculator 20180912.1

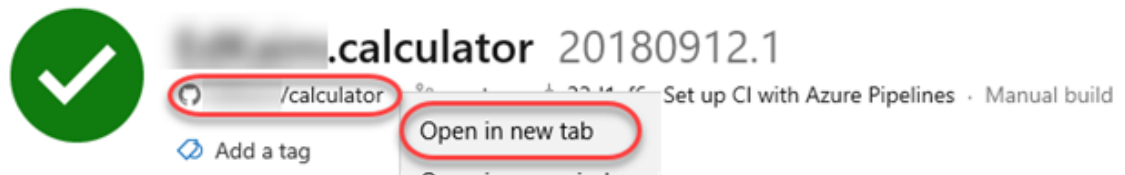
/calculator · master · 22d1ef6 : Set up CI with Azure Pipelines · Manual build

Add a tag

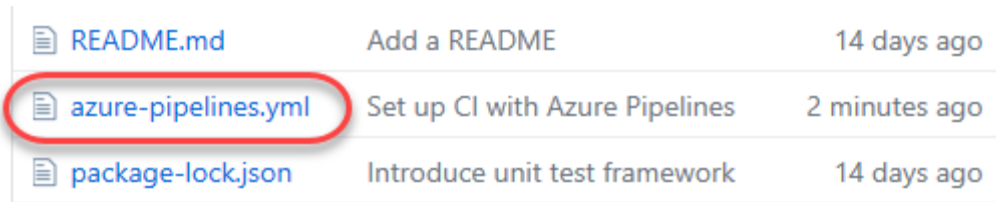
Task 3: Modifying a YAML build pipeline definition

1. While the default pipeline is a great start, it doesn't do everything we would like to have automated. For example, it would be great if it also ran our tests to confirm that the changes don't create bugs. Let's return to GitHub where we can edit the YAML by hand. Right-click the GitHub project link and select **Open in new tab**.

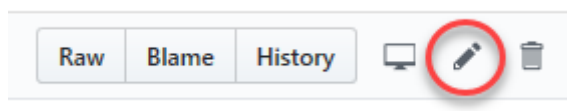
Since this lab will involve stepping back and forth between GitHub and Azure DevOps, it'll be easier to keep a browser tab open to each.



2. Open **azure-pipelines.yml**.



3. Click the **Edit** button.



4. Our project already contains tests written using Mocha so we just need to execute them in our pipeline. To add the test run, add the **"npm test"** command below. Also update the **displayName** to **'npm install, build, and test'** so that it's easier to track what each task of the build is doing later on. These are the lines to add.

5. `npm test`
 6. `displayName: 'npm install, build, and test'`
- ```
15 - script: |
16 npm install
17 npm run build
18 npm test
19 displayName: 'npm install, build, and test'
```

7. Scroll to the bottom of the page, provide some documentation for the change and click **Commit changes**. Again, it's okay to commit this change directly to the master branch for the purposes of this lab.

## Commit changes

Run 'npm test'

Run 'npm test' as part of the Azure Pipelines CI build

- ☒ Commit directly to the `master` branch.
- ☐ Create a **new branch** for this commit and start a p

Commit changes



Cancel

8. Return to the **Azure DevOps** browser tab. Use the breadcrumb navigation to return to the **Pipelines** page.



Pipelines

9. A new build should already be there. Click it to view progress.

| History                                                                                                 | Analytics*                                                                                       | Edit | Queue | ... |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|------|-------|-----|
| Commit                                                                                                  | Build #                                                                                          |      |       |     |
|  CI build for GitHub |  20180912.2 |      |       |     |

10. Depending on how quickly you got here, the build may be queued, in progress, or already done. Click **Logs** and follow it through to completion.

**Logs** Summary Tests Release Edit Queue ...

11. Once completed, click the **npm install, build, and test** task to view its log output.

✓ Get sources · succeeded

✓ Install Node.js · succeeded

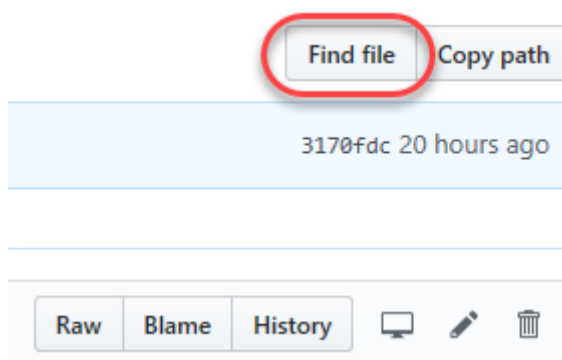
✓ npm install, build, and test · succeeded

12. After the install and build steps we can see the logs for the tests. Everything passes!  
Click **Esc** to close the task view.

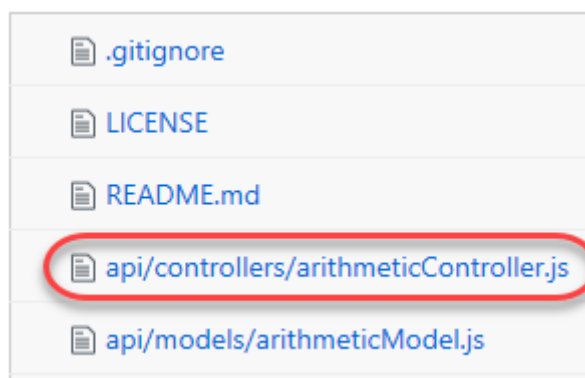
```
24 2018-09-12T21:13:29.4911733Z
25 2018-09-12T21:13:29.4941512Z Arithmetic
26 2018-09-12T21:13:29.4960373Z Validation
27 2018-09-12T21:13:29.5324003Z ✓ rejects missing operation
28 2018-09-12T21:13:29.5378551Z ✓ rejects invalid operation
29 2018-09-12T21:13:29.5425965Z ✓ rejects missing operand1
30 2018-09-12T21:13:29.5468066Z ✓ rejects missing operand2
31 2018-09-12T21:13:29.5505166Z ✓ rejects operands with invalid sign
32 2018-09-12T21:13:29.5544918Z ✓ rejects operands with invalid decimals
33 2018-09-12T21:13:29.5563826Z Addition
34 2018-09-12T21:13:29.5583595Z ✓ adds two positive integers
```

#### Task 4: Proposing a change via GitHub pull request

1. One of the great benefits of this pipeline setup is that we now have a quality gate that's automatically run every time someone commits a change. This makes it much easier to manage a project that could have any number of contributions coming in at various levels of quality. Return to the **GitHub** browser tab to test it out.
2. Click **Find file**.



3. Open **arithmeticController.js**.



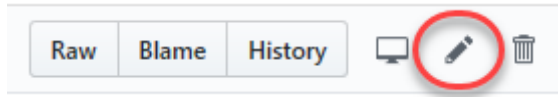
4. This controller contains the core functionality of the app. However, the code for the **add** operation isn't completely clear. Put yourself in the position of someone with good intentions, but a lack of experience with JavaScript. They might identify this as an opportunity to help out by cleaning up the code to make it better.

```

13 var operations = {
14 'add': function(a,b) { return +a + +b },
15 'subtract': function(a,b) { return a - b },
16 'multiply': function(a,b) { return a * b },
17 'divide': function(a,b) { return a / b }

```

- Click the **Edit** button.



- Remove the first and third plus signs from the **add** method to make the code easier to read. (No spoilers, please.)

```

13 var operations = {
14 'add': function(a,b) { return a + b },
15 'subtract': function(a,b) { return a - b },
16 'multiply': function(a,b) { return a * b },
17 'divide': function(a,b) { return a / b }

```

- Scroll down and add some documentation for the change. Also select **Create a new branch** and give it the name **"addition-cleanup"**. Click **Propose file change**.

Update addition

Removed unnecessary plus signs

☐ Commit directly to the `master` branch.

☒ Create a new branch for this commit and

addition-cleanup

Propose file change Cancel

- Click **Create pull request** to kick off the process of getting your untested changes into some production code!

Update addition

Write Preview

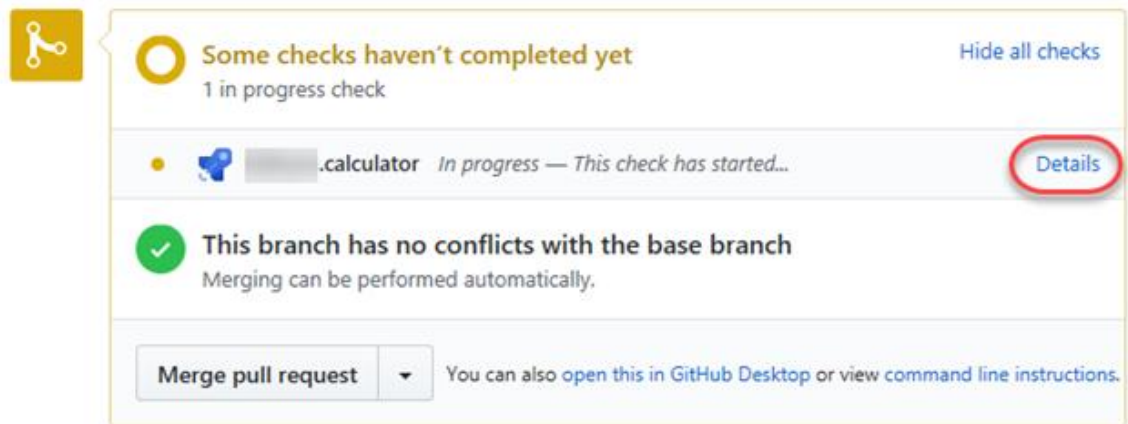
Removed unnecessary plus signs

Attach files by dragging & dropping or [selecting them](#).

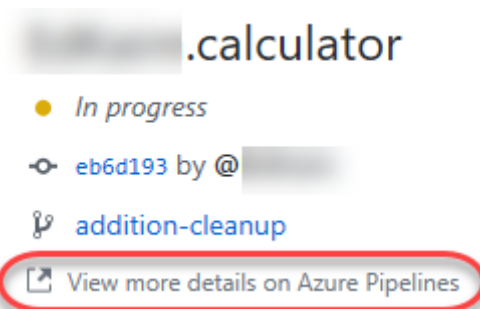
Styling with Markdown is supported

Create pull request

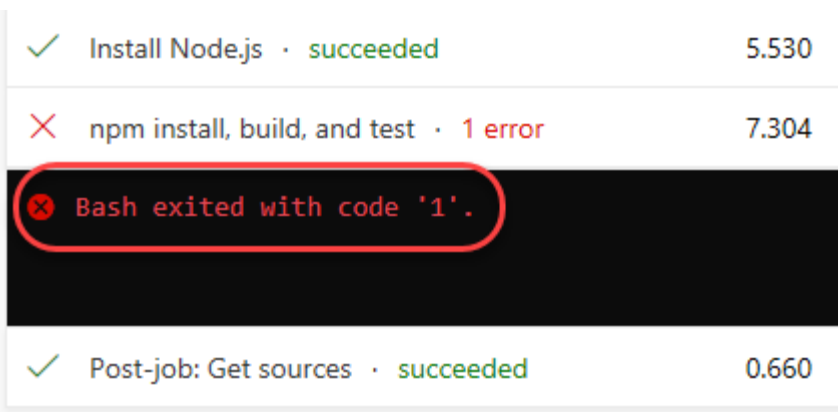
9. Azure DevOps will detect the change and start the build pipeline. This will update the UI in GitHub explaining that some of the checks haven't completed yet. Return to your original mindset of "project owner" and click **Details** to learn more.



10. Click **View more details on Azure Pipelines**. This will open a new tab.



11. Follow the build until it fails. Click the **npm install, build, and test** task to view the log output.



12. Locate the section that lists out failing tests. It might not be immediately clear why the tests failed, but all of the history we've accrued in the pipeline makes it easy to identify that something from this new pull request is the cause. The next step will be to figure out why "21 + 21" produced "2121" instead of the expected "42".

```


62 2018-09-13T16:53:17.325029Z 6 failing
63 2018-09-13T16:53:17.325640Z
64 2018-09-13T16:53:17.327133Z 1) Arithmetic
65 2018-09-13T16:53:17.328659Z Addition
66 2018-09-13T16:53:17.330172Z adds two positive integers:
67 2018-09-13T16:53:17.330867Z
68 2018-09-13T16:53:17.332816Z Uncaught AssertionError: expected { result: '2121' } to deeply equal { result: 42 }
69 2018-09-13T16:53:17.335275Z + expected - actual
70 2018-09-13T16:53:17.336508Z
71 2018-09-13T16:53:17.338079Z {
72 2018-09-13T16:53:17.339770Z - "result": "2121"
73 2018-09-13T16:53:17.341754Z + "result": 42
74 2018-09-13T16:53:17.343136Z }
75 2018-09-13T16:53:17.344598Z
76 2018-09-13T16:53:17.346008Z at Test.<anonymous> (test/arithmic.js:58:35)
77 2018-09-13T16:53:17.347336Z at Test.assert (node_modules/supertest/lib/test.js:181:6)
78 2018-09-13T16:53:17.348787Z at Server.assert (node_modules/supertest/lib/test.js:131:12)
79 2018-09-13T16:53:17.350290Z at emitCloseNT (net.js:1664:8)
80 2018-09-13T16:53:17.351766Z at _combinedTickCallback (internal/process/next_tick.js:136:11)
81 2018-09-13T16:53:17.353186Z at process._tickCallback (internal/process/next_tick.js:181:9)

```

13. Close the current tab.

### Task 5: Using the broken pull request to improve the project

1. Return to the **GitHub** browser tab and the role of project owner.
2. Click the commit to view its details. (Your code will appear different to the following screenshot)

 .calculator

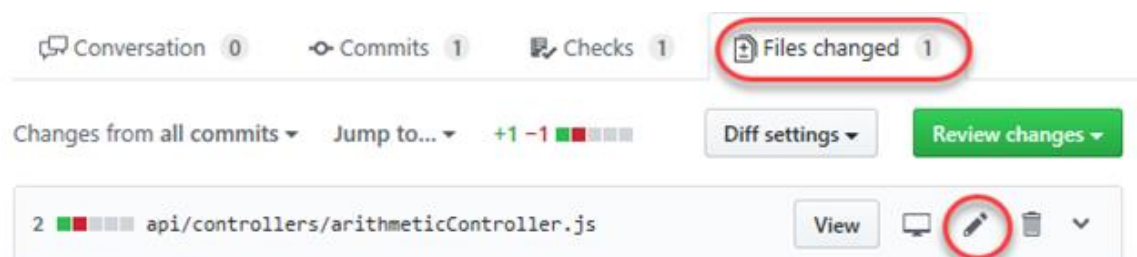
✖ Failure Re-run

🕒 built 5 minutes ago in less than a minute

🔑 c7b2668 by @

🔗 addition-cleanup

3. Select the **Files changed** tab and click the **Edit** button.



4. It would appear that the changes were made by someone who didn't realize that the plus signs before each variable were necessary to coerce those variables to their number representations. By removing them, JavaScript interpreted the middle plus sign as the string concatenation operator, which explains why  $21 + 21 = 2121$  in the failed test. Undo the original changes by adding the plus signs before the **a** and **b** variables. Also add a comment explaining that this is necessary for the operation to perform as expected.

```
13 var operations = {
14 // Using + operator to coerce variables to numbers to avoid string concatenation.
15 'add': function(a,b) { return +a + +b },
16 'subtract': function(a,b) { return a - b },
}
```

5. Scroll down to document the changes and click **Commit changes**.

### Commit changes

Added comment to explain addition

Code is efficient, but confusing.

☒ Commit directly to the `addition-cleanup` branch.

☐ Create a new branch for this commit and start a pull

Commit changes

Cancel

6. Select the **Conversation** tab.


Conversation 0


Commits 2

Checks 0


Files changed 1


7. Azure DevOps will again detect the change and start the build pipeline. Wait for the following section to change to green to indicate that all checks have passed.



**Some checks haven't completed yet**Hide all checks

1 in progress check

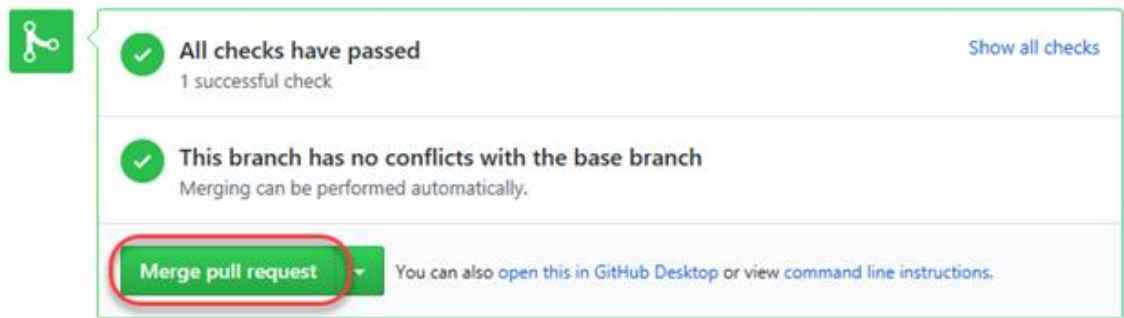
`.calculator` *In progress — This check has started...*Details

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

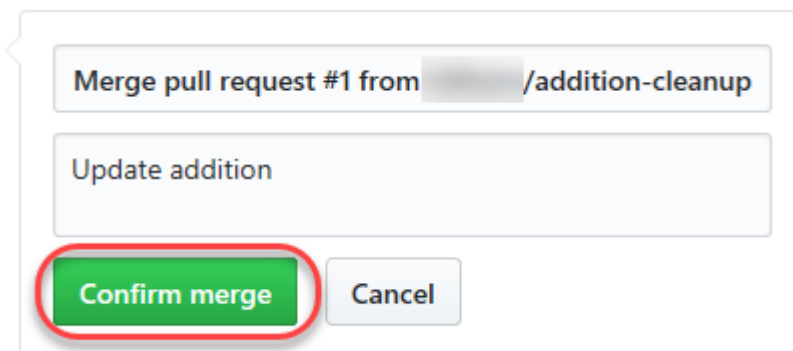
Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

8. Once all checks have passed, click **Merge pull request**.

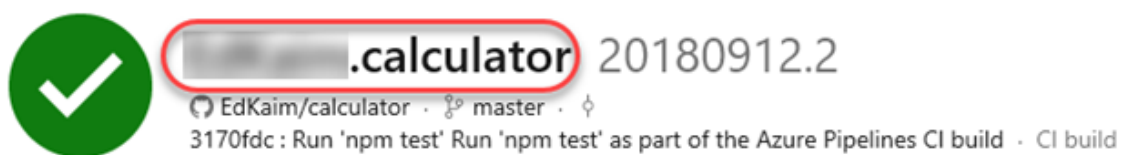


9. Click **Confirm merge**.



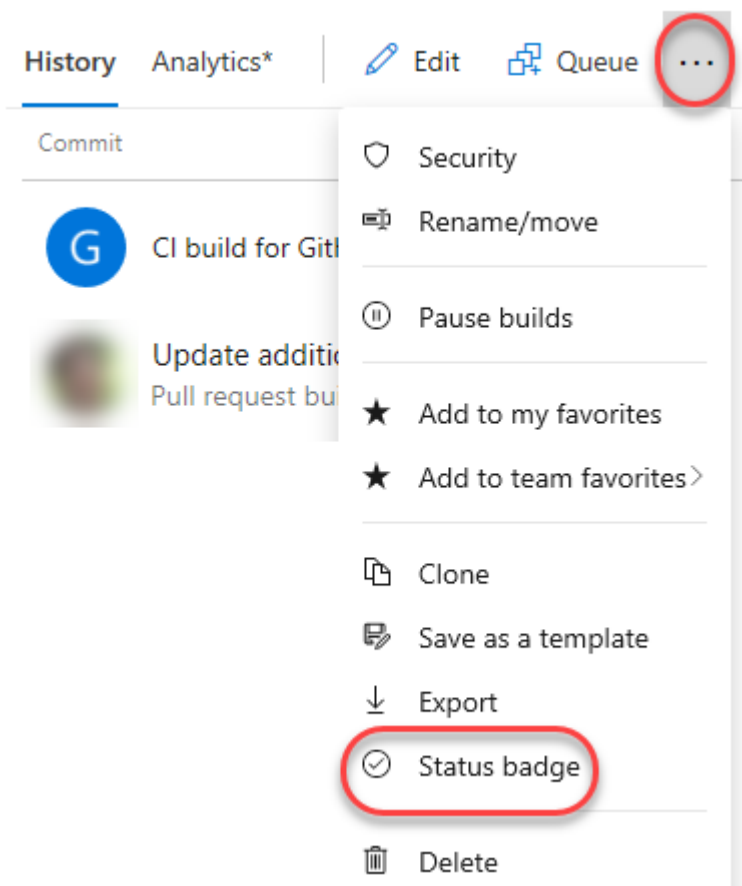
### ***Task 6: Adding a build status badge***

1. An important sign for a quality project is its build status badge. When someone finds a project that has a badge indicating that the project is currently in a successful build state, it's a sign that the project is maintained effectively. Return to the **Azure DevOps** tab.
2. Click the build pipeline to navigate to its overview page.



3. From the **ellipses** dropdown, select **Status badge**.





4. The **Status badge** UI provides a quick and easy way to integrate the build status wherever you want. Often, you'll want to use the provided URLs in your own dashboards, or you can use the Markdown snippet to add the status badge to locations such as Wiki pages. Click the **Copy to clipboard** button for **Sample Markdown**.

## Status badge



### Image URL

[https://dev.azure.com/\[redacted\]/\\_apis/build/status/\[redacted\].c...](https://dev.azure.com/[redacted]/_apis/build/status/[redacted].c...)



### Default branch URL

[https://dev.azure.com/\[redacted\]/\\_apis/build/status/\[redacted\].c...](https://dev.azure.com/[redacted]/_apis/build/status/[redacted].c...)

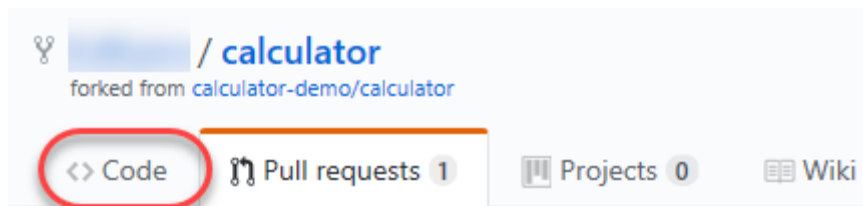


### Sample Markdown

`[[Build Status]](https://dev.azure.com/\[redacted\]/\_apis/build/s...)`



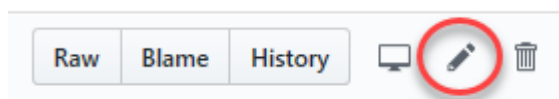
5. Return to the **GitHub** tab.
6. Select the **Code** tab.



7. Open **README.md**.

|            |                                                   |                |
|------------|---------------------------------------------------|----------------|
| api        | Added comment to explain addition                 | 10 minutes ago |
| public     | Add a user-interface                              | 15 days ago    |
| test       | Introduce arithmetic tests                        | 15 days ago    |
| .gitignore | calculator: REST API to perform simple arithmetic | 15 days ago    |
| LICENSE    | calculator: REST API to perform simple arithmetic | 15 days ago    |
| README.md  | Add a README                                      | 15 days ago    |

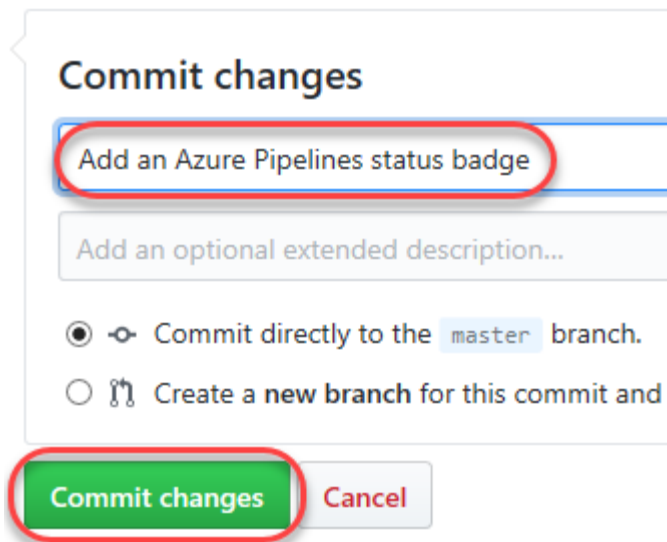
8. Click the **Edit** button.



9. Paste in the clipboard contents around line 5.

```
1 Calculator
2 *****
3 A simple node.js project that behaves like a pocket calculator.
4
5 [![Build Status](https://dev.azure.com/[redacted]/_apis/build/status/[redacted].calculator)]
6 (https://dev.azure.com/[redacted]/_build/latest?definitionId=41)
7
8 The project contains a simple node.js application that exposes REST APIs
 to perform arithmetic on integers, and provides a test suite with mocha
```

10. Scroll down and add a commit comment and click **Commit changes**.



11. You now have a dynamic build status badge on your project's front page that allows everyone to know that you're effectively managing your project.

## Calculator

A simple node.js project that behaves like a pocket calculator.



The project contains a simple node.js application that exposes test suite with mocha and chai. The `mocha-junit-reporters` package is in

### Summary

In this lab, you learned how to integrate a GitHub project with Azure DevOps using the new Azure Pipelines integration from the Marketplace.